MANUAL DEL PROGRAMADOR

1. INTRODUCCIÓN

Cualquier portal web utiliza algún sistema de base de datos para almacenar la información que luego mostrará en sus páginas. Para ello es necesario ejecutar una serie de acciones básicas para su buen funcionamiento, que quedan referenciadas bajo el acrónimo de **CRUD**: Create, Read, Update y Delete.
Para desarrollar esta práctica en Laravel vamos a generar dos **CRUDs** mediante el **crud-generator** para que cree de forma rápida la estructura base en nuestro proyecto.
Éstos se llamarán "**Tablageneral"** y "**Tablausuario"** que serán tablas de nuestra base de datos interrelacionadas.
A partir de aquí, también añadiremos un sistema de autenticación con UI de Bootstrap.

2. REQUISITOS DEL SISTEMA

a. **XAMPP** → Servidor local
b. **Composer** → Gestor de dependencias php
c. **Node.js** → Paquete de librerías en javascript

3. CREACIÓN DEL PROYECTO

a. Crear base de datos en nuestro gestor PHPMyAdmin de Xampp con el nombre **"practicalaravel"**.

b. En el archivo **.env**, cambiamos el nombre `DB_DATABASE` en nuestro proyecto Laravel, poniendo el nombre que hemos creado en el paso anterior (practicalaravel), dejando `DB_USERNAME` como "**root"** y el campo `DB_PASSWORD` **vacío**

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=practicalaravel
DB_USERNAME=root
DB_PASSWORD=
```

c. Mediante la consola, acceder a la carpeta htdocs de Xampp y ejecutar los siguientes comandos:

i. C:\xampp\htdocs> `composer create-project laravel/laravel proyectoFinal`
Siendo *"proyectoFinal"* el nombre que le queremos dar al proyecto.

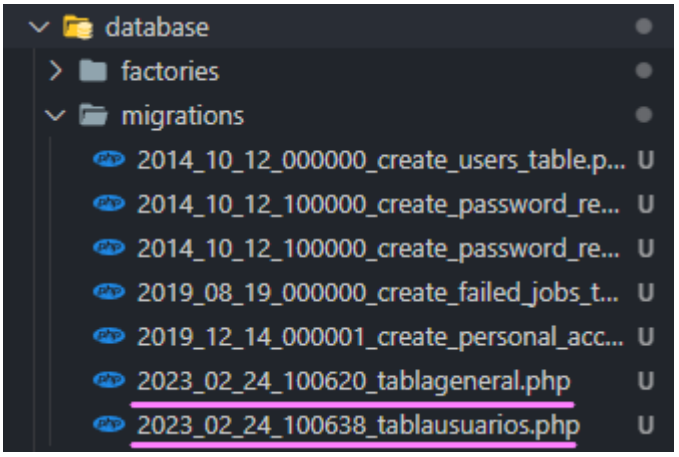
d. Nos colocamos en la carpeta del proyecto que acabamos de crear e intentamos ejecutarlo para ver que se ha creado correctamente:
   i. C:\xampp\htdocs\proyectoFinal> `php artisan serve`

4. CREACIÓN DE LAS MIGRACIONES

a. Cosas a tener en cuenta
   i. El **orden** de creación de las tablas, es decir, crear primero las tablas que **no contienen** campos relacionales.
   ii. En nuestro ejemplo las tablas serán **"tablageneral"** y **"tablausuarios"**.

b. Una vez sabemos qué tabla no contiene campos relacionales **la creamos primero** de la siguiente manera; en la cmd, nos situamos en nuestro proyecto y ejecutamos los siguientes comandos:
   i. C:\xampp\htdocs\proyectoFinal> `php artisan make:migration tablageneral`
   ii. C:\xampp\htdocs\proyectoFinal> `php artisan make:migration tablausuarios`

c. En nuestro IDE entramos en **database\migrations** y buscamos los archivos que contengan los nombres de nuestras tablas:



Una vez situados en estos archivos crearemos las tablas con sus respectivos campos:

i. Tabla general:

```php
public function up(): void
{
    Schema::create('tablageneral', function (Blueprint $table) {
        $table->engine='InnoDB';
        $table->id();
        $table->string('DNI');
        $table->timestamps();
    });
}
```

ii. Tabla usuarios:

```php
public function up(): void
{
    Schema::create('tablausuarios', function (Blueprint $table) {
        $table->engine='InnoDB';
        $table->id();
        $table->bigInteger('idusuario')->unsigned();
        $table->string('DNI');
        $table->string('nombre');
        $table->string('apellidos');
        $table->integer('numcuenta');
        $table->timestamps();
        $table->foreign('idusuario')->references('id')->on('tablageneral')->onDelete('cascade');
    });
}
```

En el caso de la tabla que contenga referencias habrá que indicarle qué campo (`→foreign('idusuario')`) hace referencia a qué campo (`→references('id')`) de qué tabla (`→on('tablageneral')`).

iii. Para añadir que en caso de querer borrar una categoría se haga un borrado en cascada, (`→onDelete('cascade')`) habrá que indicarle que **las tablas** utilizará motor InnoDB (`$table→engine='InnoDB';`).

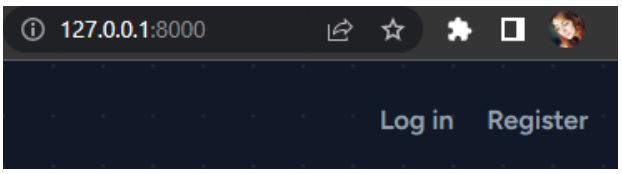
5. CREACIÓN DE LA AUTENTICACIÓN

Para la creación de la autenticación utilizaremos **ui** con **bootstrap**.
Los comandos que tenemos que ejecutar son los siguientes, **en orden**:

a. C:\xampp\htdocs\proyectoFinal> `composer require laravel/ui`
b. C:\xampp\htdocs\proyectoFinal> `php artisan ui bootstrap --auth`
c. C:\xampp\htdocs\proyectoFinal> `npm install`
d. C:\xampp\htdocs\proyectoFinal> `npm run dev`

Éste último comando lo debemos **dejar ejecutando** en un terminal **antes** de ejecutar nuestro proyecto.

Una vez terminados estos pasos, al ejecutar nuestro proyecto desde la consola debería aparecer en la parte superior derecha de la página de bienvenida los dos nuevos elementos "**Login**" y "**Register**".



Laravel crea **automáticamente** la tabla ***users*** al crear el proyecto, de manera que no nos tenemos que preocupar por guardar los usuarios.

6. CREACIÓN DE LOS CRUDs CON CRUD-GENERATOR

Para generar los CRUDs necesarios para nuestro proyecto ejecutaremos los siguientes comandos **en orden**, como siempre, desde la carpeta de nuestro proyecto:

a. C:\xampp\htdocs\proyectoFinal> `composer require ibex/crud-generator --dev`
b. C:\xampp\htdocs\proyectoFinal> `php artisan vendor:publish --tag=crud --auth`
c. C:\xampp\htdocs\proyectoFinal> `php artisan make:crud tablageneral`
d. C:\xampp\htdocs\proyectoFinal> `php artisan make:crud tablausuarios`

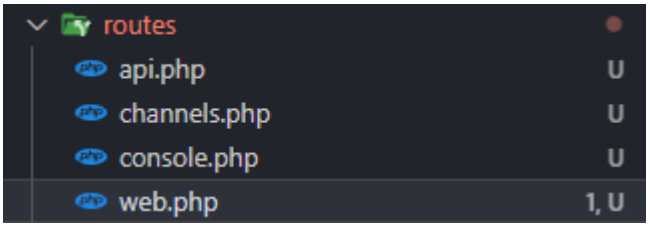
Cosas a tener en cuenta:

i. Los nombres que usemos tienen que coincidir con el nombre de **la tabla** que queramos usar para el CRUD.
ii. Estos comandos nos generan automáticamente los modelos, controladores y vistas básicas.

7. ACCESO A LOS CRUDS

Para acceder a los CRUDs que hemos generado anteriormente, debemos establecer una ruta a cada uno de ellos.
Para establecer las rutas seguiremos los siguientes pasos:

a. Nos dirigimos a la carpeta **routes** y al archivo **web.php:**



b. Establecemos las rutas a nuestros nuevos Controllers con la clase **resource** añadiendo `→middleware('auth');` para que sólo se puedan visualizar **sólo** en caso de estar registrado:

```
Route::resource('tablageneral', TablageneralController::class)->middleware('auth');
Route::resource('tablausuarios', TablausuarioController::class)->middleware('auth');
```

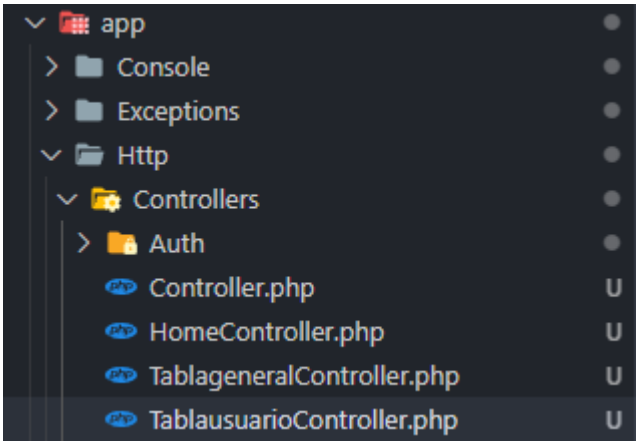
c. Cosas a tener en cuenta:
Si no se importa automáticamente, tendremos que implementar los uses necesarios:

```
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\TablageneralController;
use App\Http\Controllers\TablausuarioController;
```

8. ACCESO A LOS DATOS DE LOS USUARIOS DESDE LA TABLA GENERAL

Para acceder a los datos de la tabla general desde la tabla de usuarios hay que modificar los métodos de la tabla de usuarios para incluir los datos de la tabla general.
Para ello seguiremos los siguientes pasos:

a. Acceder a la carpeta **app\Controllers\TablausuarioController.php**



b. Agregar el **use** a Tablageneral al principio del controlador:

```
use App\Models\Tablageneral;
```

c. Crear una variable `$categorias` variable que almacenará **en orden** los campos que nosotros le indiquemos como parámetros en el método `pluck` de la siguiente manera:

i. En el método `create()`:

```php
public function create()
{
    $tablausuario = new Tablausuario();
    $tablageneral = Tablageneral::pluck('id','id');
    return view('tablausuario.create', compact('tablausuario', 'tablageneral'));
}
```

ii. En el método `edit($id)`:

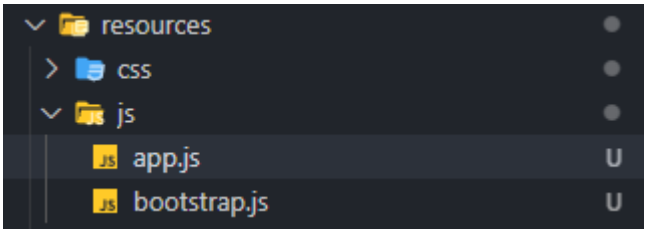
```php
public function edit($id)
{
    $tablausuario = Tablausuario::find($id);
    $tablageneral = Tablageneral::pluck('id','id');
    return view('tablausuario.create', compact('tablausuario', 'tablageneral'));
}
```

9. MODIFICACIÓN DEL TEMA POR DEFECTO

Para modificar el tema por defecto hay que seguir los siguientes pasos desde la carpeta de nuestro proyecto:

a. Instalar el package de **Bootswatch** vía **npm** mediante el siguiente comando:
C:\xampp\htdocs\proyectoFinal> `npm install bootswatch`

b. Acceder a la carpeta **app\resources\js\app.js:**



c. Añadir el siguiente import **en la primera línea** del documento:
`import "bootswatch/dist/[tema]/bootstrap.min.css";`
dónde [tema] será cambiado por el tema elegido.
Se pueden ver en la página de Bootswatch